IN THE CLAIMS:

Please cancel claims 6, 20, and 37, and amend claims 7, 21, 38, 65, 68, 69, 73 and 74 as follows:

Claim 1. (original) A method comprising:
employing a computer for:
obtaining a collection of code;
providing a program graph representing said collection of code;
identifying any authorization resources of said collection of code;
locating any bounded path within said program graph; and
associating said any authorization resource with said any bounded path.

Claim 2. (original) A method as recited in claim 1, wherein said collection of code includes codes obtained from a group of codes including basic blocks, class methods, classes, collections of classes or any combination of these.

Claim 3. (original) A method as recited in claim 1, wherein the step of providing includes constructing said program graph through static analysis techniques.

Claim 4. (original) A method as recited in claim 3, wherein the step of constructing includes employing object code.

Claim 5. (original) A method as recited in claim 1, wherein the step of identifying includes finding at least one authorization point in said program graph.

Claim 6. (canceled)

Claim 7. (currently amended) A method as recited in claim ~~6~~ 5, wherein the step of finding the ~~Java~~ authorization point includes finding a AccessController.checkPermission node in said program graph, and finding a java.security.Permission object passed as an argument to the AccessController.checkPermission method.

Claim 8. (original) A method as recited in claim 5, wherein said authorization point is an instruction invocation.

Claim 9. (original) A method as recited in claim 8, wherein said instruction invocation is used in a particular language for said collection of code.

Claim 10. (original) A method as recited in claim 9, wherein said particular language is C#.

Claim 11. (original) A method as recited in claim 1, wherein the step of identifying includes employing data flow analysis.

Claim 12. (original) A method as recited in claim 11, wherein the step of employing includes generating a data flow from said program graph.

Claim 13. (original) A method as recited in claim 1, wherein the step of identifying any bounded path includes locating a set of start nodes in said program graph, and locating a stop node in said program graph; and said bounded path includes all nodes within the graph bound by said start nodes and said stop node.

Claim 14. (original) A method as recited in claim 1, wherein the step of associating includes associating and aggregating said any authorization resource with said collection of code.

Claim 15. (original) An apparatus comprising:

computing means having:

means for obtaining a collection of code;

means for providing a program graph representing said collection of code;

means for identifying any authorization resources of said collection of code;

means for locating any bounded path within said program graph; and

means for associating said any authorization resource with said any bounded path.

Claim 16. (original) An apparatus as recited in claim 15, wherein said collection of code includes codes obtained from a group of codes including basic blocks, class methods, classes, collections of classes or any combination of these.

Claim 17. (original) An apparatus as recited in claim 15, wherein the means for providing includes means for constructing said program graph through static analysis techniques.

Claim 18. (original) An apparatus as recited in claim 17, wherein the means for constructing includes employing object code.

Claim 19. (original) An apparatus as recited in claim 15, wherein the means for identifying includes means for finding at least one authorization point in said program graph.

Claim 20. (canceled)

Claim 21. (currently amended) An apparatus as recited in claim ~~20~~ 19, wherein the means for finding ~~a Java~~ the authorization point includes finding a AccessController.checkPermission node in said program graph, and finding a java.security.Permission object passed as an argument to the AccessController.checkPermission method.

Claim 22. (original) An apparatus as recited in claim 19, wherein said authorization point is an instruction invocation.

Claim 23. (original) An apparatus as recited in claim 22, wherein said instruction invocation is used in a particular language for said collection of code.

Claim 24. (original) An apparatus as recited in claim 23, wherein said particular language is C#.

Claim 25. (original) An apparatus as recited in claim 15, wherein the means for identifying includes employing data flow analysis.

Claim 26. (original) An apparatus as recited in claim 25, wherein the means for employing includes generating a data flow from said program graph.

Claim 27. (original) An apparatus as recited in claim 15, wherein the means for identifying any bounded path includes locating a set of start nodes in said program graph, and locating a stop node in said program graph; and said bounded path includes all nodes within the graph bound by said start nodes and said stop node.

Claim 28. (original) An apparatus as recited in claim 15, wherein the means for associating includes associating and aggregating said any authorization resource with said collection of code.

Claim 29. (original) A method comprising:

employing a computer including the steps of:

obtaining a collection of code;

providing a program graph representing said collection of code; and

identifying a complete set of authorization resources of said collection of code.

Claim 30. (original) A method as recited in claim 29, if no resource is identified in said step of identifying, further comprising providing an indication that authorization testing is not necessary.

Claim 31. (original) A method as recited in claim 29, further comprising:

identifying any bounded path within said program graph; and

associating any authorization resource identified in the step of identifying with said any bounded path.

Claim 32. (original) An apparatus comprising:

a computer having access to a collection of code, the computer including:

an authorization resource identifier to identify any authorization resources within the collection of code;

a bounded path locator to locate any bounded path within a program graph of said collection of code; and

an associator to associate said any authorization resource with said any bounded path.

Claim 33. (original) An apparatus as recited in claim 32, wherein said collection of code includes codes obtained from a group of codes including basic blocks, class methods, classes, collections of classes or any combination of these.

Claim 34. (original) An apparatus as recited in claim 32, further comprising a program graph constructor to construct said program graph through static analysis techniques.

Claim 35. (original) An apparatus as recited in claim 32, wherein the program graph constructor uses object code of said collection of code.

Claim 36. (original) An apparatus as recited in claim 32, wherein the authorization resource identifier finds at least one authorization point in said program graph.

Claim 37. (canceled)

Claim 38. (currently amended) An apparatus as recited in claim ~~37~~ 36, wherein the ~~Java~~ authorization point includes finding a AccessController.checkPermission node in said program graph, and finding a java.security.Permission object passed as an argument to the AccessController.checkPermission method.

Claim 39. (original) An apparatus as recited in claim 36, wherein said authorization point is an instruction invocation.

Claim 40. (original) An apparatus as recited in claim 39, wherein said instruction invocation is used in a particular language for said collection of code.

Claim 41. (original) An apparatus as recited in claim 40, wherein said particular language is C#.

Claim 42. (original) An apparatus as recited in claim 32, wherein the authorization resource identifier employs data flow analysis.

Claim 43. (original) An apparatus as recited in claim 42, wherein the data flow analysis is generated from said program graph.

Claim 44. (original) An apparatus comprising:

a computing module having access to a collection of code, the computer module including:

an authorization resource identifier to completely identify any authorization resources within a collection of code; and

a bounded path locator to locate any bounded path within a program graph of said collection of code.

Claim 45. (original) An apparatus as recited in claim 44, wherein the authorization resource identifier provides an indication that authorization testing is not necessary if no resource is identified by said authorization resource identifier.

Claim 46. (original) An apparatus as recited in claim 44, further comprising an associator to associate said any authorization resource with said any bounded path.

Claim 47. (original) An apparatus comprising:

a computer including:

means for obtaining a collection of code;

means for providing a program graph representing said collection of code; and

means for identifying a complete set of authorization resources of said collection of code.

Claim 48. (original) An apparatus as recited in claim 47, further comprising means for indicating if authorization testing is necessary.

Claim 49. (original) An apparatus as recited in claim 47, further comprising:

means for identifying any bounded path within said program graph; and

means for associating any authorization resource identified in the step of identifying with said any bounded path.

Claim 50. (previously presented) A method comprising:

employing a computer for:

constructing a program graph from a collection of code using static analysis techniques;

performing a data flow analysis using static analysis techniques;

searching said program graph for any resource for executing said collection of code;

identifying any bounded path within said program graph over which said resource is utilized; and

associating said resource with said collection of code.

Claim 51. (original) The method as recited in claim 50, wherein the step of constructing includes employing source code of said collection of code.

Claim 52. (original) A method as recited in claim 50, wherein the step of constructing includes building an invocation graph of said collection of code to form said program graph.

Claim 53. (original) A method as recited in claim 50, wherein the step of constructing a program graph includes constructing a call graph of said collection of code to form said program graph.

Claim 54. (original) A method as recited in claim 50, wherein said step of constructing includes using context-sensitivity.

Claim 55. (original) A method as recited in claim 54, wherein said step of using context sensitivity includes using type information for any method receiver and/or any parameter.

Claim 56. (original) A method as recited in claim 55, wherein the step of using type information includes using class and memory allocation site information.

Claim 57. (original) A method as recited in claim 56, wherein the step of using type information includes using per instance information.

Claim 58. (original) A method as recited in claim 57, wherein the step of using instance information includes associating instance information with a node or edge in said program graph.

Claim 59. (original) A method as recited in claim 50, wherein said collection of code includes codes constructed from a group of codes including basic blocks, class methods, classes, collections of classes or any combination of these.

Claim 60. (previously presented) A method as recited in claim 50, wherein the step of searching includes locating a node or edge in said program graph that represents a location where said resource would be utilized.

Claim 61. (previously presented) A method as recited in claim 60, wherein said resource is a resource identifier.

Claim 62. (original) A method as recited in claim 60, wherein said location is an authorization test.

Claim 63. (original) A method as recited in claim 50, wherein said program graph represents an object oriented program.

Claim 64. (original) A method as recited in claim 50, wherein said program graph and said data flow analysis are constructed from Java object code.

Claim 65. (currently amended) A method as recited in claim 61, wherein said resource identifier includes at least one ~~Java~~ java.security.Permission class.

Claim 66. (original) A method as recited in claim 62, wherein said authorization test is a call to any java.security.AccessController.checkPermission method.

Claim 67. (original) A method as recited in claim 62, wherein said location represents a call to any authorization testing method in any instance of java.lang.SecurityManager and/or one of its subclasses.

Claim 68. (currently amended) A method as recited in claim 55 , wherein said node has a parameter which said type information is a ~~Java~~ java.security.Permission class.

Claim 69. (currently amended) A method as recited in claim 68, wherein the step of identifying includes locating the constructor for said ~~Java~~

java.security.Permission class allocation site and using said data flow analysis in identifying any value passed by any parameter to said constructor, wherein the combination of the Java java.security.Permission and a value for any parameter is the used Permission.

Claim 70. (original) A method as recited in claim 50, wherein the step of identifying any bounded path includes locating a set of start nodes in said program graph, and locating a stop node in said program graph; and said bounded path includes all nodes within the graph bound by said start nodes and said stop node.

Claim 71. (previously presented) A method as recited in claim 50, wherein the step of associating includes mapping a subset of said collection of code to said resource.

Claim 72. (original) A method as recited in claim 71, wherein the subset of said collection of code includes nodes in said bounded path.

Claim 73. (currently amended) A method as recited in claim 72, further comprising employing ~~a~~ privileged ~~Java~~ code wherein said stop node represents the method java.security.AccessController.checkPermission(); and

employing said start nodes are any of root nodes in said program graph or a node representing the method java.security.AccessController.doPrivileged().

Claim 74. (currently amended) A method as recited in claim 73, further comprising connecting a used Permission with said privileged ~~Java~~ code.

Claim 75. (original) A method as recited in claim 74, further comprising connecting a used Permission with any node in said program graph prior to said java.security.AccessController.doPrivileged() node.

Claim 76. (original) A method as recited in claim 74, wherein said step of associating includes connecting a used Permission for each java.security.AccessController.checkPermission() in said program graph.

Claim 77. (original) A method as recited in claim 76, wherein said step of associating includes connecting said used Permission from each node in said program graph to each method associated with said program graph.

Claim 78. (original) A method as recited in claim 77, wherein said step of associating includes connecting said used Permission from each method in said program graph to each class in said program graph.

Claim 79. (original) A method as recited in claim 78, wherein said step of associating includes connecting said used Permission from each class in said program graph to collection of classes.

Claim 80. (original) A method as recited in claim 79, wherein said step of associating includes connecting said used Permission is associated from each class in said program graph to collection of classes.

Claim 81. (previously presented) A method as recited in claim 50, further comprising employing said resource in executing said collection of code.

Claim 82. (previously presented) A method comprising:

statically detecting resources for a collection of code written in a computer programming language, by including the steps of:

calculating if any code of said collection of code is part of a program graph;

identifying any resource for said collection of code;

determining any bounded path of nodes within said program graph which constrain said resources of said collection of code; and

associating said any resource with said collection of code within said bounded path of nodes in said program graph.

Claim 83. (original) A method as recited in claim 82, wherein the step of calculating includes using an invocation graph having a set of root nodes for said program graph.

Claim 84. (original) A method as recited in claim 82, wherein the step of calculating includes using a call graph having a set of root nodes for said program graph.

Claim 85. (original) A method as recited in claim 83, further comprising determining whether a basic block, method, class is represented as a node in said invocation graph.

Claim 86. (original) A method as recited in claim 84, further comprising determining whether a basic block, method, class is represented as a node in said call graph.

Claim 87. (previously presented) A method as recited in claim 83, further comprising identifying a node in said invocation graph that identifies said resources, wherein said resources is one or more resources.

Claim 88. (previously presented) A method as recited in claim 84, further comprising a step identifying a node in said invocation graph that identifies said resources, wherein said resources is one or more resources.

Claim 89. (original) A method as recited in claim 83, wherein nodes in said bounded path of nodes are all nodes in the graph that are bounded between start nodes and a stop node.

Claim 90. (original) A method as recited in claim 84, wherein nodes in said bounded path of nodes are all nodes in the graph that are bounded between a set of start nodes and a stop node.

Claim 91. (original) A method as recited in claim 90, wherein all root nodes in said invocation graph are members of the start nodes.

Claim 92. (original) An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing association, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 1.

Claim 93. (original) An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing identification, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 29.

Claim 94. (original) An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing association, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 50.

Claim 95. (original) An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing statistical detection, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 80.

Claim 96. (original) A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing association, the computer readable program code means in said

computer program product comprising computer readable program code means for causing a computer to effect the functions of the elements in claim 15.

Claim 97. (original) A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing association, the computer readable program code means in said computer program product comprising computer readable program code means for causing a computer to effect the functions of the elements in claim 32.

Claim 98. (original) A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing identification, the computer readable program code means in said computer program product comprising computer readable program code means for causing a computer to effect the functions of the elements in claim 44.

Claim 99. (original) A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing identification, the computer readable program code means in said computer program product comprising computer readable program code means for causing a computer to effect the functions of the elements in claim 47.

Claim 100. (original) A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for identification, said method steps comprising the steps of claim 1.